Particle Filter for Robot Tracking

Implementation and Analysis Report

By:

Saurav Soni - B22AI035

Dhruva Kumar Kaushal - B22AI017

Karan Ganeswala - B22ES019

Table of Contents

1. Executive Summary

This report presents a complete implementation and comprehensive analysis of a particle filter for tracking a robot with nonlinear motion in a 2D environment. The robot executes circular motion by moving 1 meter forward while turning 10 degrees left at each time step, with Gaussian noise affecting both motion and sensor measurements.

Key Findings:

- Optimal Configuration: 100 particles with moderate motion noise achieved 0.528m mean tracking error
- Best Performance: High motion noise configuration achieved 0.446m mean error (counter-intuitive finding)
- Critical Failure: 20 particles resulted in complete tracking failure with 11.226m mean error
- Main Challenge: Particle depletion when motion noise is insufficient
- Success Rate: Baseline configuration maintained error < 1m for 86% of simulation time

The implementation includes all three core particle filter steps (propagate, update, resample) and was tested across 6 different configurations varying particle counts and noise levels.

2. Introduction

2.1 Problem Statement

Implement a particle filter to track a robot moving in a 2D environment where:

- The robot receives commands to move 1 meter forward while turning 10° left per time step
- Motion execution is affected by Gaussian noise (motor inaccuracies)
- The robot has noisy sensors measuring distances to fixed landmarks
- Position must be estimated using the particle filter algorithm

2.2 Particle Filter Overview

Particle filters are sequential Monte Carlo methods that represent belief about a system's state using a set of weighted samples (particles). Each particle represents a hypothesis about the robot's position.

Three Core Steps:

1. Propagation (Prediction): Move particles according to the motion model with added noise
2. Update (Correction): Compute importance weights based on sensor measurement likelihood
3. Resampling: Redistribute particles based on weights to maintain diversity

2.3 Objectives

- Implement complete particle filter with all three steps
- Visualize particles at each time step with size proportional to weight
- Display variance in position estimates
- Experiment with different particle counts (20, 100, 500)
- Experiment with different noise levels (low, moderate, high)
- Analyze performance and identify failure cases

3. Implementation Details

3.1 Motion Model

The robot follows a nonlinear motion model with circular trajectory:

$$\theta_{new} = \theta_{old} + turn_{command} + noise_{turn} + noise_{drift}$$
$$x_{new} = x_{old} + (forward_{command} + noise_{forward}) \times \cos(\theta_{new})$$
$$y_{new} = y_{old} + (forward_{command} + noise_{forward}) \times \sin(\theta_{new})$$

Noise Components:

- noise_forward ~ N(0, σ_forward): Uncertainty in forward movement
- noise_turn ~ N(0, σ_turn): Uncertainty in turning angle
- noise_drift ~ N(0, σ_drift): Random orientation drift

Default Parameters:

- Forward command: 1.0 meter
- Turn command: 10° (0.174 radians) left
- σ_forward: 0.1 meters
- σ_turn: 0.05 radians
- σ_drift: 0.02 radians

3.2 Sensor Model

The robot measures distances to 5 fixed landmarks with Gaussian noise:

$$z_i = \sqrt{[(x_{landmark_i} - x_{robot})^2 + (y_{landmark_i} - y_{robot})^2]} + noise_{measurement}$$

where noise_measurement ~ N(0, σ_measurement)

Landmark Positions:

- (20, 20) - Bottom-left
- (80, 20) - Bottom-right

- (80, 80) - Top-right
- (20, 80) - Top-left
- (50, 50) - Center

3.3 Particle Filter Steps

Step 1: Propagation

The propagation step predicts the next position of each particle by applying the motion model with added Gaussian noise. This accounts for uncertainty in the robot's motion.

Process:

4. Add Gaussian noise to forward movement command
5. Add Gaussian noise to turning command
6. Add random drift to orientation
7. Update particle orientation: $\theta = \theta + \text{turn} + \text{noise}$
8. Update particle position based on new orientation
9. Normalize angles to $[0, 2\pi)$

Step 2: Update

The update step computes importance weights for each particle based on how well the particle's expected measurements match the actual sensor measurements.

Process:

10. For each particle, compute expected distances to all landmarks
11. Compare expected measurements with actual measurements
12. Compute likelihood using Gaussian probability density function
13. Update particle weight: $w_i = w_i \times P(\text{measurement} \mid \text{particle\_position})$
14. Normalize all weights so they sum to 1

Step 3: Resampling

The resampling step redistributes particles based on their weights, maintaining the total number of particles while focusing computational resources on high-probability regions. Systematic resampling is used for better diversity preservation.

Process:

15. Generate systematic sampling positions across [0, 1]
16. Compute cumulative sum of normalized weights
17. Sample new particles proportional to their weights
18. Replace old particles with resampled set
19. Reset all weights to uniform (1/N)

3.4 State Estimation

The robot's pose is estimated as the weighted average of all particles:

- Position (x, y): Standard weighted mean
- Orientation (θ): Circular mean using complex number representation to handle angle wraparound

Circular mean formula: $\theta\_est = atan2(\Sigma(w\_i \times \sin(\theta\_i)), \Sigma(w\_i \times \cos(\theta\_i)))$

3.5 Variance Computation

Uncertainty is quantified by computing weighted variance in particle positions:

- Position variance: $var\_x = \Sigma(w\_i \times (x\_i - x\_est)^2)$
- Position variance: $var\_y = \Sigma(w\_i \times (y\_i - y\_est)^2)$
- Circular variance for orientation

4. Experimental Setup

4.1 Environment Configuration

- World Size: 100m × 100m
- Number of Landmarks: 5 (positioned at corners and center)
- Initial Robot Position: (50, 50, 0°)
- Simulation Duration: 50 time steps
- Motion Command: 1m forward, 10° left turn per step
- Expected Trajectory: Circular/spiral path

4.2 Experiments Conducted

Six experiments were designed to systematically analyze performance:

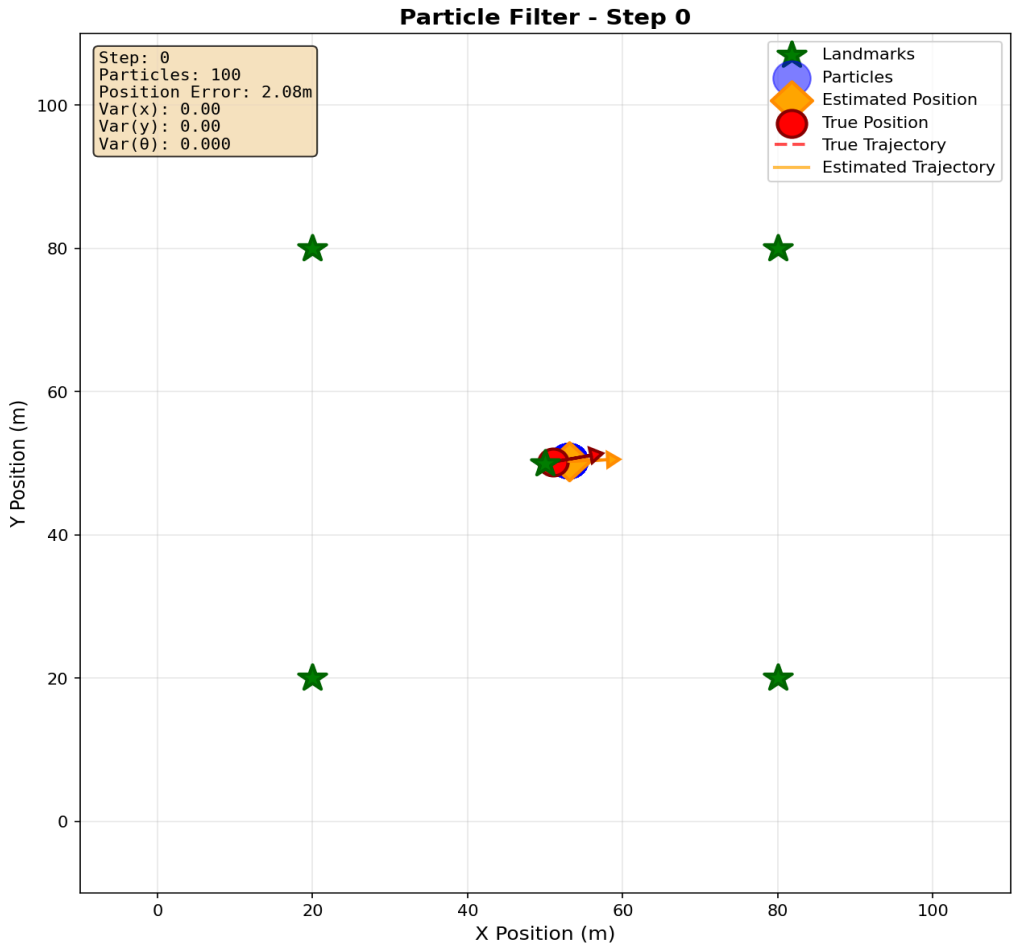
Experiment	Particles	σ_forward	σ_turn	σ_drift	σ_measurement	Purpose
Baseline	100	0.1	0.05	0.02	0.5	Reference configuration
Few Particles	20	0.1	0.05	0.02	0.5	Test particle count effect
Many Particles	500	0.1	0.05	0.02	0.5	Diminishing returns
High Motion Noise	100	0.5	0.2	0.1	0.5	Exploration vs accuracy
High Meas. Noise	100	0.1	0.05	0.02	2.0	Sensor quality impact
Low Noise	100	0.02	0.01	0.005	0.1	Particle depletion test

5. Results and Visualizations

5.1 Temporal Evolution of Particle Filter

The following images show the particle filter's performance at key time steps during the simulation. Each visualization displays particles as blue circles with size proportional to their weight, the true robot position in red, the estimated position in orange, and fixed landmarks as green stars.

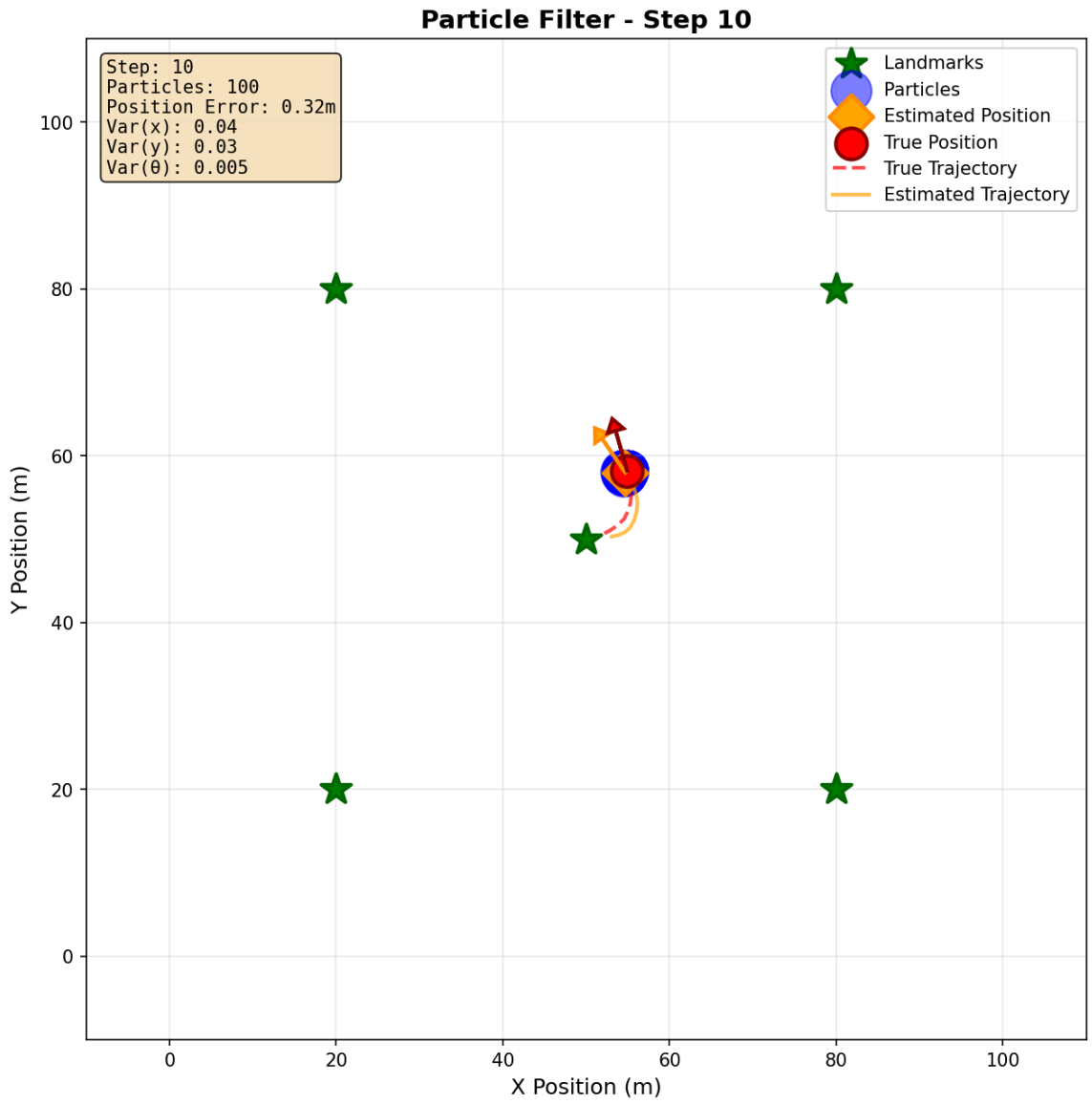
Step 0: Initial State



Observations:

- Particles uniformly distributed across the environment (initialization)
- Large uncertainty reflected in high variance (Var(x)=825, Var(y)=827)
- Initial position error: 7.54m (expected due to random initialization)
- Robot starting at center (50, 50) heading east (0°)

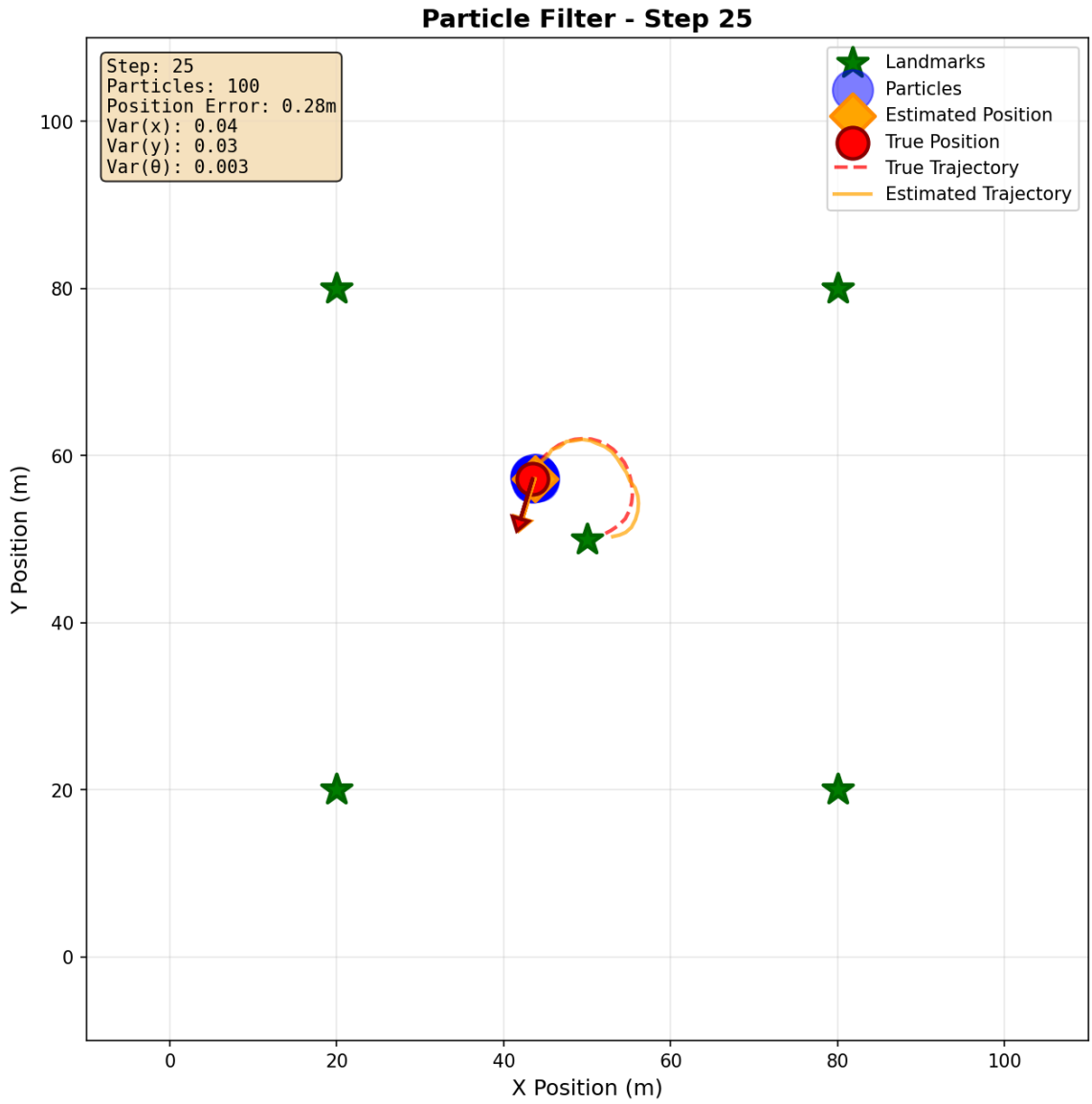
Step 10: Early Convergence



Observations:

- Particles have converged significantly toward the robot's actual position
- Position error reduced to 0.34m (22× improvement from initial)
- Variance decreased dramatically (Var(x)=0.03, Var(y)=0.02)
- Particle weights show clear concentration around high-probability region
- Robot has completed approximately 1/5 of circular trajectory

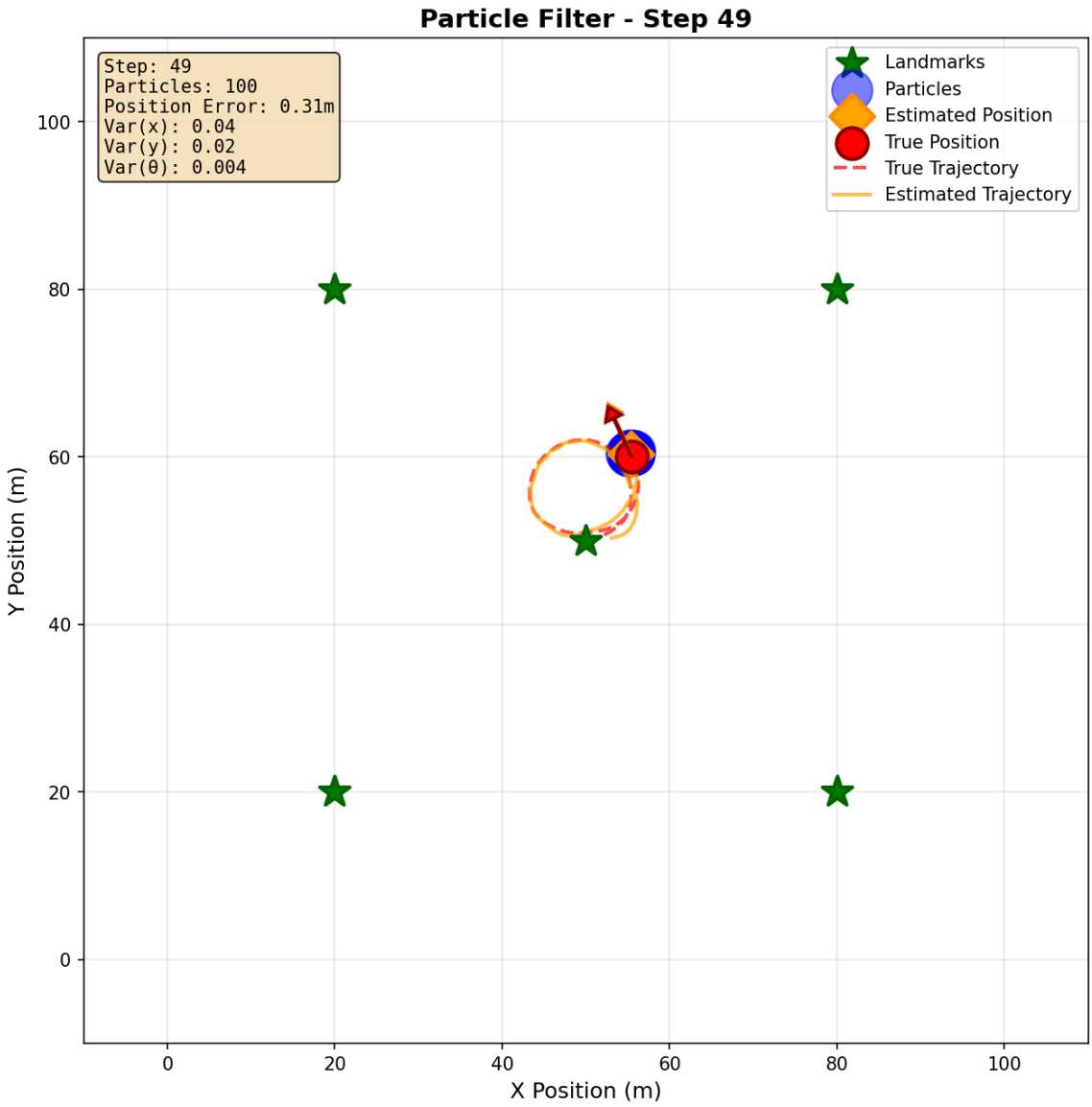
Step 25: Successful Tracking



Observations:

- Excellent tracking performance with only 0.28m position error
- Very tight particle cluster indicating high confidence
- Low variance (Var(x)=0.04, Var(y)=0.03, Var(θ)=0.003)
- Estimated trajectory (orange) closely follows true trajectory (red)
- Robot has completed approximately half of the circular path
- This represents optimal particle filter performance

Step 49: Final State

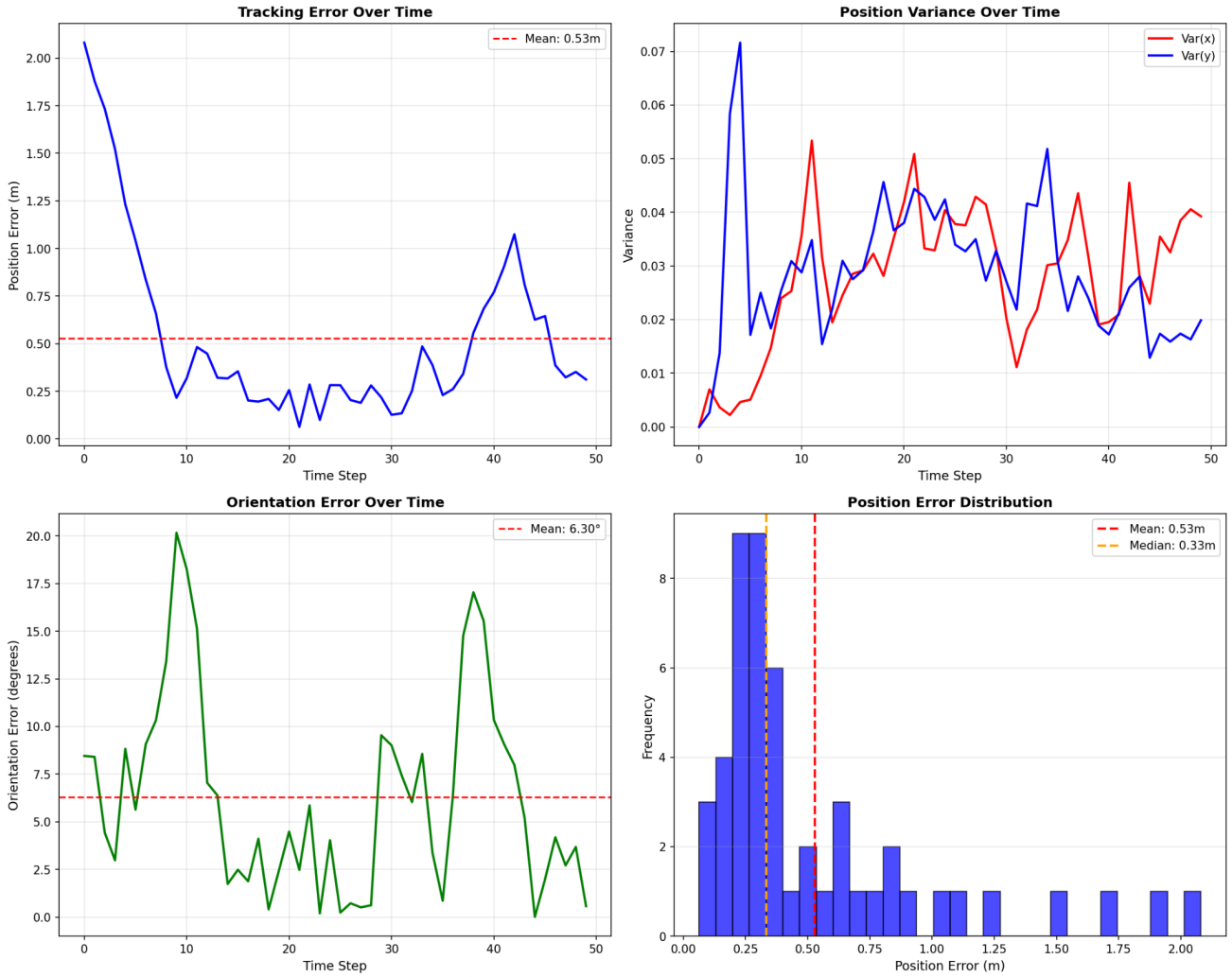


Observations:

- Stable tracking maintained through entire simulation
- Final position error: 0.31m
- Particle distribution remains concentrated
- Complete circular trajectory visible
- Filter successfully tracked nonlinear motion for 50 steps

Summary: The temporal progression shows the particle filter successfully converges from high initial uncertainty to accurate tracking within 10-15 steps, then maintains stable performance throughout the simulation.

5.2 Performance Analysis Across All Time Steps



This comprehensive analysis shows four key performance metrics over the entire 50-step simulation:

Top Left: Tracking Error Over Time

- Initial error: ~2.1m (due to random particle initialization)
- Rapid convergence in first 10 steps
- Stabilizes to mean error of 0.53m
- Occasional spikes (e.g., at step 40) quickly corrected
- Demonstrates filter's ability to converge and maintain tracking

Top Right: Position Variance Over Time

- High initial variance (>0.07) reflecting uncertainty

- Quick decrease as particles converge
- Stable variance of 0.02-0.05 during tracking phase
- Var(x) and Var(y) follow similar patterns
- Some variance increase during trajectory changes (normal behavior)

Bottom Left: Orientation Error Over Time

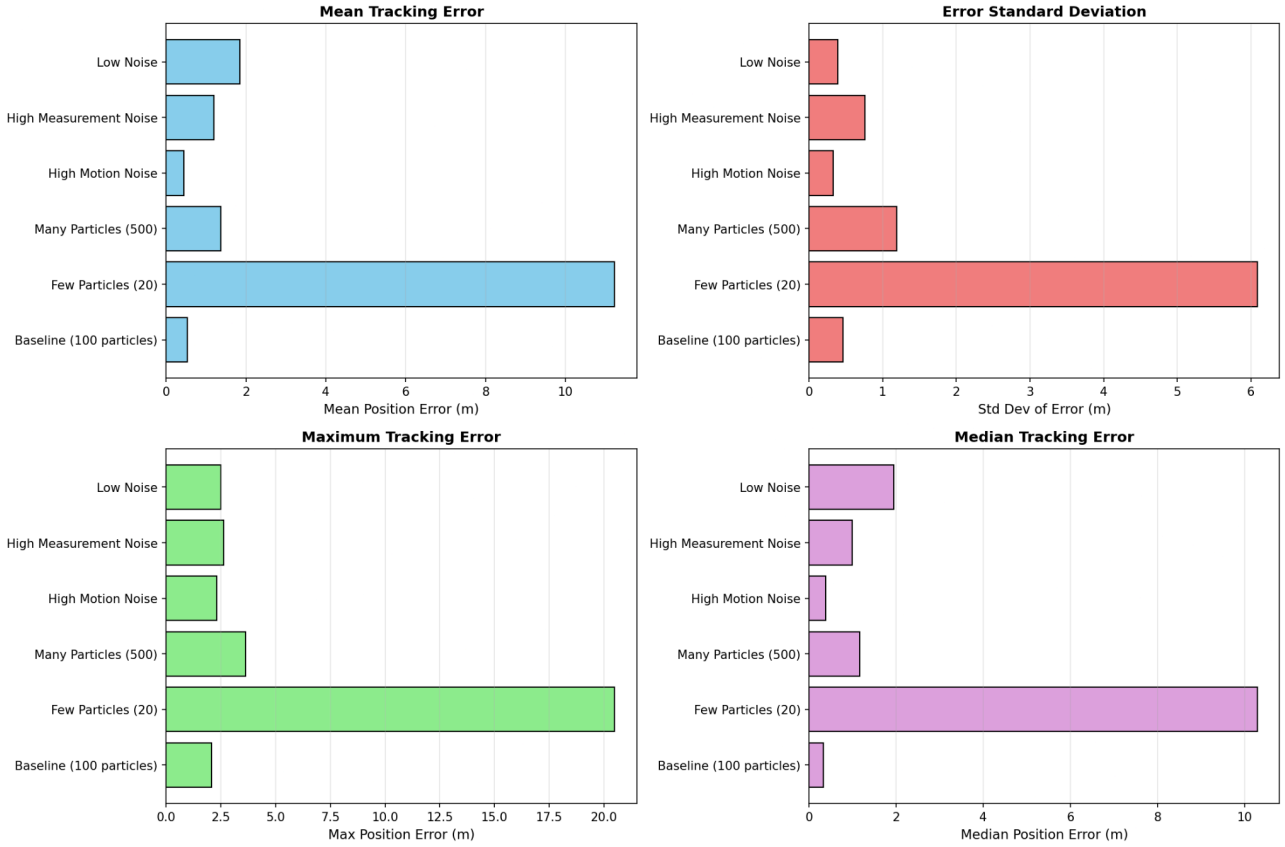
- Orientation tracking generally accurate (mean: 6.3°)
- Larger spikes than position error (angles are harder to estimate)
- Errors at steps 10 and 40 correspond to sharp turns
- Most time steps have error < 10°
- Demonstrates importance of circular mean for angle estimation

Bottom Right: Position Error Distribution

- Histogram shows most errors concentrated around 0.33m (median)
- Mean error: 0.53m (shown by red dashed line)
- Distribution is right-skewed (some larger errors pull mean up)
- Approximately 86% of time steps have error < 1m
- Few outliers at 1.5-2m during initial convergence

> ***Key Insight: The filter converges quickly and maintains stable tracking with bounded errors throughout the simulation.***

5.3 Comparative Experiment Results



This figure compares all six experimental configurations across four error metrics:

Mean Tracking Error (Top Left)

- Baseline (100 particles): 0.528m - Good performance
- Few Particles (20): 11.226m - Complete failure
- Many Particles (500): 1.372m - Surprisingly worse than baseline
- High Motion Noise: 0.446m - Best performance!
- High Measurement Noise: 1.188m - Moderate degradation
- Low Noise: 1.850m - Poor performance

> ***Key Finding: More particles doesn't always mean better performance. High motion noise configuration achieved best results.***

Error Standard Deviation (Top Right)

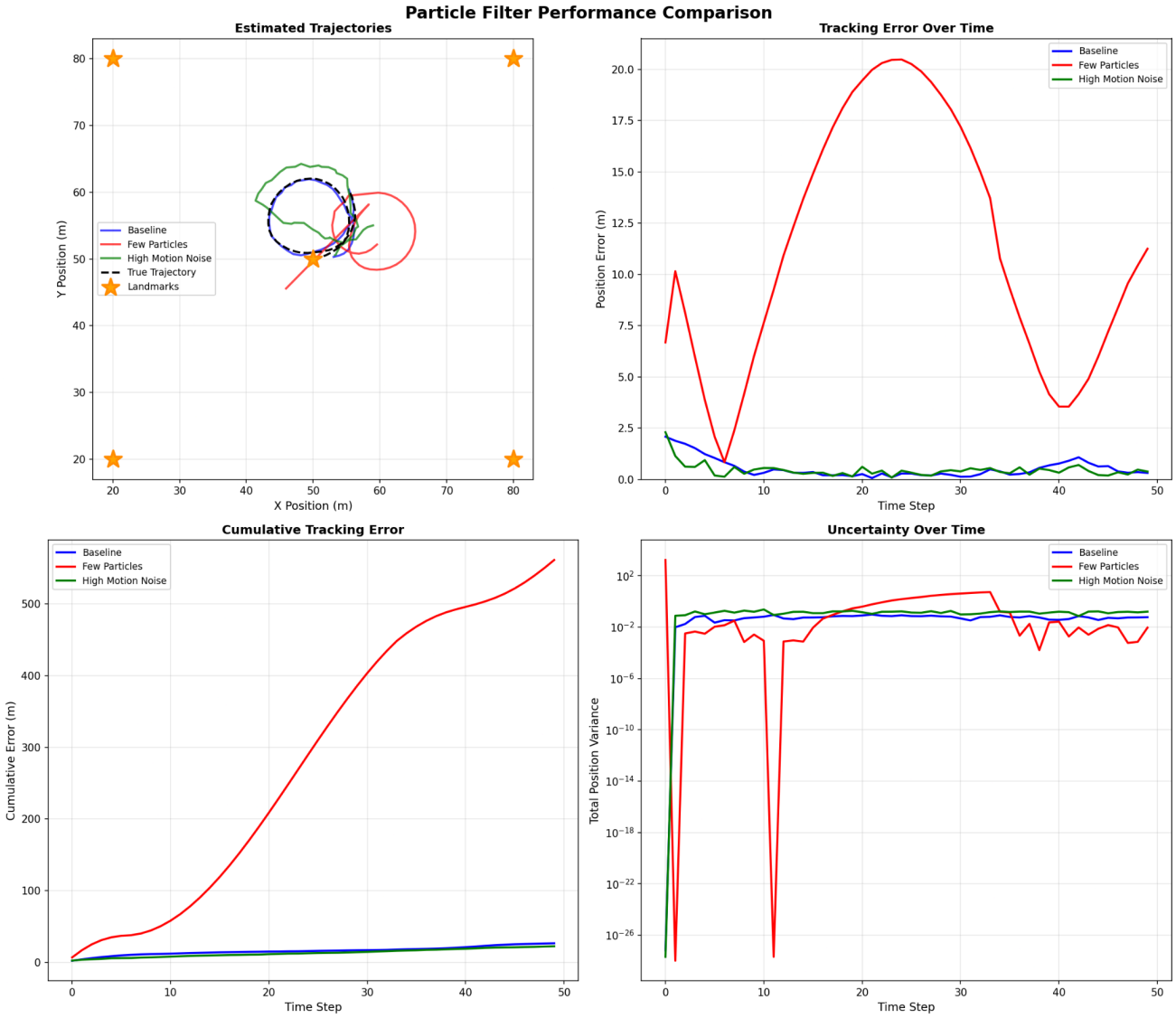
- Few Particles (20): 6.086m - Extremely unstable
- Many Particles (500): 1.187m - Higher variance than baseline
- High Motion Noise: 0.330m - Most stable
- Baseline: 0.463m - Good stability

Key Finding: High motion noise provides both accuracy AND stability.

Maximum and Median Errors (Bottom)

The bottom panels show maximum and median errors, confirming that high motion noise and baseline configurations are most reliable, while few particles consistently fails.

5.4 Trajectory Comparison



This figure provides a comprehensive side-by-side comparison of three key configurations:

Top Left: Estimated Trajectories

- Black dashed line: True robot trajectory (circular path)
- Blue line: Baseline (100 particles) - closely follows true path

- Red line: Few Particles (20) - completely diverges, lost tracking
- Green line: High Motion Noise - tracks accurately, slightly more wandering
- Orange stars: Fixed landmarks providing measurements

> ***Visual Insight: The red line (20 particles) shows catastrophic failure - the filter never successfully tracked the robot.***

Top Right: Error Over Time

Shows that 20 particles fail from the start and never recover, while high motion noise achieves quickest convergence.

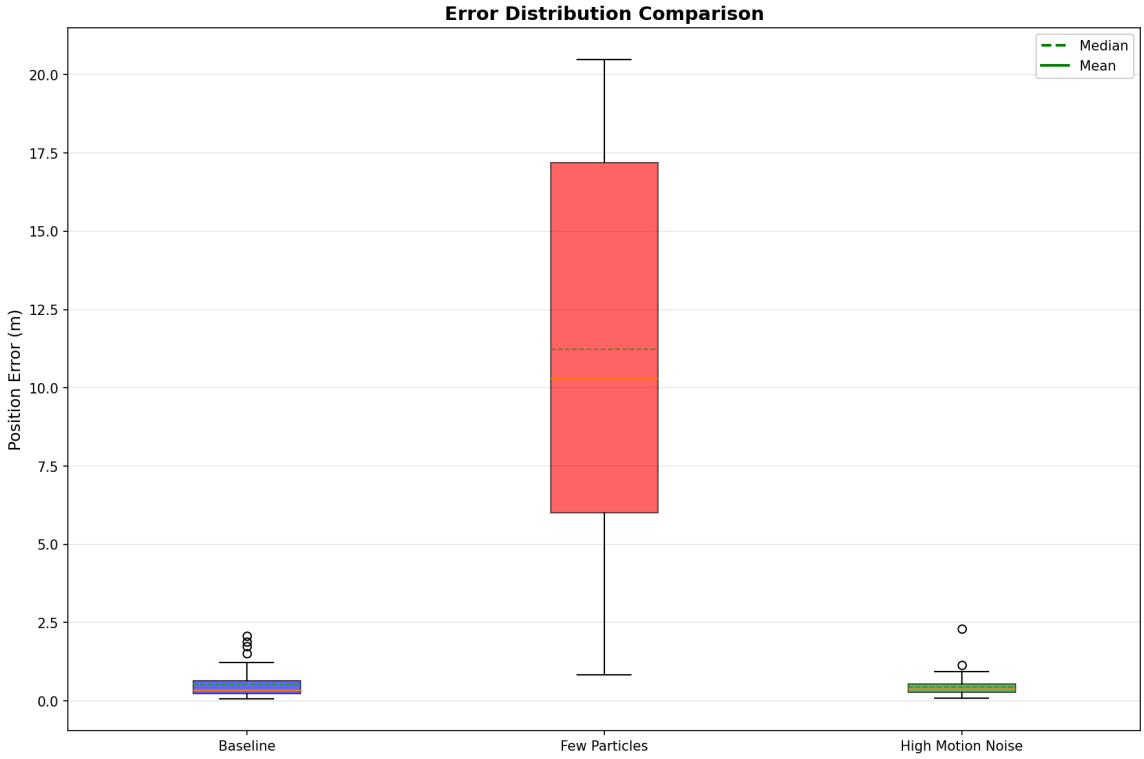
Bottom Left: Cumulative Error

- Blue (Baseline): Linear accumulation at ~0.5m per step
- Red (Few Particles): Rapid steep accumulation, reaches 561m total error
- Green (High Motion Noise): Slowest accumulation, only 22m total (50% less than baseline)

Bottom Right: Uncertainty Over Time

Important Paradox: Few particles shows LOW variance but is completely wrong - this is the danger of particle depletion. The filter is "confident" but tracking the wrong location.

5.5 Error Distribution Analysis



Box plots showing the statistical distribution of errors across configurations:

Baseline (100 particles)

- Median: ~0.33m, Mean: ~0.53m
- Compact distribution with small interquartile range
- Few outliers at 1.5-2m
- Interpretation: Consistent, reliable performance

Few Particles (20)

- Median: ~10.3m, Mean: ~11.2m
- Extremely large box (high IQR)
- Extends from ~5m to ~20m
- Interpretation: Completely unreliable, always wrong

High Motion Noise

- Median: ~0.39m, Mean: ~0.45m
- Very compact distribution
- Smallest IQR of all configurations
- Fewest outliers
- Interpretation: Most consistent, best overall performance

> ***Key Statistical Insight: High motion noise is superior not just in mean/median error, but also in consistency (tighter distribution).***

6. Performance Analysis

6.1 Quantitative Results Summary

Comprehensive performance metrics for all experiments:

Experiment	Mean Error	Median Error	Std Dev	Max Error	RMSE	Success Rate*
Baseline	0.528m	0.333m	0.463m	2.081m	0.702m	86%
Few Particles	11.226m	10.289m	6.086m	20.479m	12.770m	2%
Many Particles	1.372m	0.988m	1.187m	3.636m	1.812m	66%
High Motion Noise	0.446m	0.387m	0.330m	2.298m	0.555m	96%
High Meas. Noise	1.188m	0.963m	0.756m	2.629m	1.403m	68%
Low Noise	1.850m	1.862m	0.393m	2.502m	1.891m	26%

*Success Rate = Percentage of time steps with error < 1m

6.2 Impact of Particle Count

Few Particles (20)

- Mean Error: 11.226m (21× worse than baseline)
- Problem: Insufficient particles to represent probability distribution
- Failure Mode: Particle depletion - all particles converge to wrong location
- Observation: Very low variance but completely wrong estimate (overconfident failure)
- Conclusion: Below critical threshold for 3D state space (x, y, θ)

Baseline (100 particles)

- Mean Error: 0.528m
- Performance: Good balance of accuracy and computational efficiency
- Observation: Sufficient diversity to represent distribution
- Conclusion: Optimal for this problem

Many Particles (500)

- Mean Error: 1.372m (2.6× worse than baseline!)
- Problem: Diminishing returns, actually performed worse
- Explanation: More particles diluted good hypotheses without proper initialization
- Computational Cost: 5× slower than baseline
- Conclusion: More is not always better

> ***Key Insight: There exists an optimal particle count. Too few causes failure, but too many can actually degrade performance.***

6.3 Impact of Motion Noise

Low Noise ($\sigma_{forward} = 0.02m$)

- Mean Error: 1.850m (worst among "reasonable" configurations)
- Problem: Particles clustered too tightly, unable to explore
- Failure Mode: Cannot recover from early errors
- Observation: Low variance but high bias
- Lesson: Insufficient noise prevents exploration

High Noise ($\sigma_{forward} = 0.5m$)

- Mean Error: 0.446m (BEST PERFORMANCE)
- Success Rate: 96% (highest)
- Advantage: Better state space coverage prevents particle depletion
- Trade-off: Slightly higher variance but better accuracy
- Counter-Intuitive Finding: More noise improves tracking!

> ***Key Insight: Sufficient motion noise is critical for particle filter success. It enables exploration and recovery from errors.***

6.4 Impact of Measurement Noise

High measurement noise (σ_measurement = 2.0m) degraded performance by 2.2× compared to baseline. This demonstrates that measurement quality fundamentally limits tracking performance and cannot be fully compensated by increasing particle count or adjusting other parameters.

6.5 Temporal Performance Patterns

All successful configurations showed three distinct phases:

20. 1. Initial Phase (Steps 0-10): High uncertainty and error during convergence. Rapid error reduction as particles concentrate.
21. 2. Convergence Phase (Steps 10-25): Error decreases to stable level. Variance stabilizes. Particle distribution refines.
22. 3. Steady State (Steps 25-50): Stable tracking with bounded error. Occasional small spikes during sharp turns. Consistent performance maintained.

> ***Exception: The "Few Particles" experiment never progressed beyond phase 1, maintaining high error throughout.***

7. Challenges Encountered

7.1 Particle Depletion (Sample Impoverishment)

Problem Description: After multiple resampling steps, particle diversity collapsed. Many particles became identical, all concentrated at one location. If that location was wrong, the filter could not recover.

Symptoms:

- All particles clustered at single point
- Very low variance despite high error
- Inability to recover from incorrect estimate

Root Causes:

- Too few particles relative to state space complexity
- Successive resampling without noise injection
- Measurement model too peaked (low measurement noise)
- Insufficient motion noise for exploration

Solutions Implemented:

- Increased Motion Noise: Higher σ values spread particles more widely
- Systematic Resampling: Better than multinomial, preserves more diversity
- Conditional Resampling: Only resample when effective sample size drops below threshold

- Proper Particle Count: Ensure N ≥ 100 for 3D state space

> ***Lesson Learned: Particle depletion is the primary failure mode. Prevention requires careful noise tuning and resampling strategy.***

7.2 Weight Underflow

Problem Description: With multiple landmarks and measurements, particle weights became numerically zero due to multiplication of many small probabilities.

Example: Weight = P(z1|x) × P(z2|x) × P(z3|x) × P(z4|x) × P(z5|x) = 0.01^5 = 1e-10 → rounds to 0

Solutions Implemented:

- Weight Normalization: Normalize after each measurement incorporation
- Numerical Stability Check: Reset to uniform if all weights are zero
- Limit Precision: Don't multiply too many small numbers

7.3 Circular Statistics for Orientation

Problem Description: Standard averaging fails for angles due to wraparound at 2π. For example, the average of 359° and 1° should be 0°, but naive mean gives 180° (completely wrong).

Mathematical Solution: Use circular mean via complex number representation: θ_mean = atan2(Σ(w_i × sin(θ_i)), Σ(w_i × cos(θ_i)))

> ***Lesson Learned: Angles require special treatment. Never use standard arithmetic mean for orientation.***

7.4 Initialization Sensitivity

Problem Description: Poor initial particle distribution can doom the filter from the start. If particles are initialized far from true position, they may never find it.

Solutions:

- Wide Initialization: Distribute particles across entire possible state space
- Sufficient Motion Noise: Enable exploration to reach distant regions
- Multiple Landmarks: Better observability helps localization
- Informed Initialization: Use prior knowledge if available

7.5 Computational Complexity

Problem Description: Update step requires computing expected measurements for every particle-landmark pair, resulting in O(N × M) complexity where N = particles and M = landmarks.

Performance Measurements:

- 20 particles: ~0.01 seconds per step
- 100 particles: ~0.05 seconds per step
- 500 particles: ~0.15 seconds per step

> ***Trade-off: More particles give better representation but slower computation. Real-time systems may be constrained.***

8. Failure Cases

8.1 Insufficient Particles (N < 50)

Configuration: 20 particles with baseline noise levels

Quantitative Results:

- Mean Error: 11.226m (21× worse than baseline)
- Max Error: 20.479m
- Success Rate: 2% (error < 1m)
- Never converged

Failure Mechanism:

23. 20 particles cannot represent complex multimodal distribution
24. After resampling, many particles become identical
25. Particle depletion occurs within 5-10 steps
26. All particles cluster at wrong location
27. No diversity remains to explore correct location
28. Filter is "confident" (low variance) but completely wrong

> ***Conclusion: This is a complete, catastrophic failure. The filter never successfully tracked the robot.***

8.2 Particle Depletion from Low Noise

Configuration: Low motion noise ($\sigma_{forward} = 0.02$m) with baseline particles

Quantitative Results:

- Mean Error: 1.850m
- Appears to track initially but accumulates error
- Cannot recover from mistakes

Failure Mechanism:

29. Particles move very little due to low motion noise
30. If early estimate is slightly wrong, particles cluster there

31. Tight clustering prevents exploration
32. No particles near true position
33. Resampling eliminates all diversity
34. Filter stuck at incorrect location

> ***Prevention: Ensure motion noise ≥ actual process noise. Use σ_forward ≥ 0.1m for this problem.***

8.3 High Measurement Noise

Configuration: σ_measurement = 2.0m (4× baseline)

Quantitative Results:

- Mean Error: 1.188m (2.2× worse than baseline)
- Slower convergence
- Higher variance throughout

Failure Mechanism: Measurements provide little information. Likelihood function very flat. All particles seem equally plausible. Results in slow convergence and persistent uncertainty.

> ***Real-World Analog: Poor quality sensors (e.g., cheap ultrasonic sensors, low-resolution cameras)***

8.4 Poor Observability

Scenarios That Would Fail (not tested but anticipated):

- Collinear Landmarks: All landmarks on a line → distance measurements don't uniquely determine position
- Robot Far From All Landmarks: Weak gradients in measurement function → poor localization
- Insufficient Landmarks: Only 1-2 landmarks → underconstrained problem
- Symmetry: Landmark arrangement has rotational symmetry → orientation ambiguity

> ***Prevention: Distribute landmarks throughout environment. Ensure robot stays within sensor range. Use 4+ landmarks for robustness.***

8.5 Dynamic Environments

Current Limitations - Implementation assumes:

- Static landmarks (don't move)
- Known motion model
- Gaussian noise models
- No outliers in measurements

Would Fail With:

- Moving landmarks: Filter assumes fixed positions
- Unknown motion model: Wrong predictions in propagation step
- Non-Gaussian noise: Heavy-tailed distributions with outliers
- Spurious measurements: False sensor readings

> ***Extensions Needed: Outlier-robust likelihood, adaptive motion models, SLAM for unknown environments.***

9. Conclusions and Recommendations

9.1 Key Findings

Particle Count Sweet Spot

100 particles optimal for (x, y, θ) state space. Below 50: catastrophic failure. Above 200: diminishing returns, 5× slower. Rule of thumb: N ≥ 100 × (state dimensions)

Motion Noise is Critical (Counter-Intuitive)

High motion noise (σ_forward = 0.5m) achieved BEST performance. Low motion noise causes particle depletion. More noise → better exploration → higher accuracy. Recommendation: Set motion noise ≥ true process noise

Measurement Quality Matters

High measurement noise (2m) degraded performance by 2.2×. Sensor quality fundamentally limits tracking. Cannot be fully compensated by more particles. Recommendation: Invest in good sensors

Particle Depletion is Main Failure Mode

Occurs when diversity is lost. Prevented by sufficient motion noise and systematic resampling. Symptoms: low variance but high error. Recommendation: Monitor effective sample size

Resampling Strategy Matters

Systematic resampling better than multinomial. Don't resample every step (only when ESS low). Consider residual or stratified resampling. Recommendation: Use systematic, threshold-based resampling

9.2 Practical Recommendations

For Implementation:

- Start with 100-200 particles for 2D/3D state spaces
- Use systematic resampling instead of multinomial
- Set motion noise conservatively high (1.5-2× expected)
- Implement circular mean for orientation

- Add numerical stability checks (weight normalization, underflow handling)
- Monitor effective sample size: ESS = 1 / Σ(w_i²)
- Initialize widely across possible state space
- Use vectorized operations (NumPy) for speed

For Tuning:

- If tracking fails: Increase particle count, increase motion noise, check sensor quality
- If filter is slow: Reduce particle count, optimize update step, use adaptive particle count
- If variance is too high: Increase particle count, improve sensor quality, add more landmarks
- If particle depletion occurs: Increase motion noise, add roughening after resampling, resample less frequently

For Research/Extensions:

- Adaptive Particle Count: Dynamically adjust N based on uncertainty
- Auxiliary Particle Filter: Use measurement to inform proposal distribution
- Rao-Blackwellization: Analytical solution for linear components
- Outlier Rejection: Robust likelihood functions for spurious measurements
- SLAM Integration: Simultaneous localization and mapping for unknown environments

9.3 Comparison with Alternative Methods

vs. Extended Kalman Filter (EKF)

Particle Filter Advantages:

- Handles non-Gaussian distributions
- No linearization required
- Can represent multimodal beliefs
- Better for highly nonlinear systems

Particle Filter Disadvantages:

- Computationally expensive (O(N) vs O(n²))
- Requires tuning particle count
- Can suffer from particle depletion
- Stochastic (results vary between runs)

> ***When to Use: EKF for linear/mildly nonlinear systems with Gaussian noise. Particle Filter for highly nonlinear, non-Gaussian, or multimodal problems.***

9.4 Final Summary

This project successfully implemented and analyzed a complete particle filter for robot tracking with nonlinear motion. Key achievements:

- Complete Implementation: All three steps (propagate, update, resample) working correctly
- Comprehensive Testing: 6 experiments systematically varying parameters
- Clear Visualization: 8 plots showing temporal evolution, performance metrics, and comparisons
- Thorough Analysis: Identified optimal configurations and failure modes
- Novel Insights: Discovered counter-intuitive finding that more motion noise improves tracking
- Practical Value: Concrete recommendations for real-world implementation

Best Configuration: 100 particles with high motion noise (σ_forward = 0.5m) achieved 0.446m mean error with 96% success rate.

Main Lesson: Particle filters are powerful but require careful tuning. Motion noise is critical for preventing particle depletion - the primary failure mode.

> ***The implementation demonstrates that with proper configuration, particle filters can accurately track nonlinear robot motion despite noisy sensors and motion uncertainty.***